

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-151743

(P 2 0 0 0 - 1 5 1 7 4 3 A)

(43) 公開日 平成12年 5 月 30 日 (2000. 5. 30)

(51) Int. Cl. ⁷	識別記号	F I	テーマコード [*]	(参考)
H04L 29/06		H04L 13/00	305	Z 5B076
G06F 9/06	530	G06F 9/06	530	W 5K034

審査請求 未請求 請求項の数 6 O L (全17頁)

(21) 出願番号 特願平10-328053

(22) 出願日 平成10年11月18日 (1998. 11. 18)

(71) 出願人 000004226

日本電信電話株式会社

東京都千代田区大手町二丁目 3 番 1 号

(72) 発明者 依田 育生

東京都新宿区西新宿 3 丁目19番 2 号 日本
電信電話株式会社内

(72) 発明者 川幡 太一

東京都新宿区西新宿 3 丁目19番 2 号 日本
電信電話株式会社内

(74) 代理人 100069981

弁理士 吉田 精孝

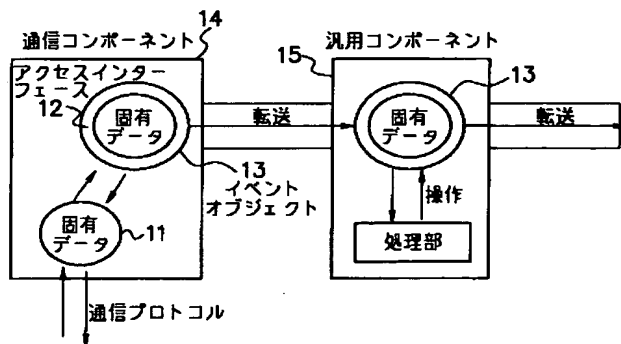
最終頁に続く

(54) 【発明の名称】 ネットワーク制御システムの設計方法

(57) 【要約】

【課題】 プロトコルの多様化に対応し得るコンポーネントウェアの組み合わせによるネットワーク制御システムの設計方法を提供すること。

【解決手段】 プロトコル固有の固有データ 11 にプロトコルによらないアクセスインターフェース 12 を実装させてイベントオブジェクト 13 を構成することによって、プロトコル固有のデータ構造を保持させたままコンポーネント 14, 15 間で交換可能とし、プロトコルによらない設計を可能とする。



【特許請求の範囲】

【請求項 1】 ネットワーク管理システムであって、ネットワークの通信装置を制御する機能要素となるコンポーネントウェアを用意・選択して、選択されたコンポーネントウェアの機能を使用する目的に合わせてカスタマイズし、このように設定されたコンポーネントウェアを幾つも接続することで設計を行うネットワーク制御システム設計方法において、

接続されたコンポーネントウェア間で、多様なデータ構造のデータ自体は異なりつつも、統一されたデータアクセスインターフェースでアクセスを可能にすることによって、コンポーネントウェアで扱うデータまたはコンポーネントウェア自身でカスタマイズするデータまたはコンポーネントウェア間でやりとりされるのをモニタするデータに対する表示及び編集の方法を統一することを特徴とするネットワーク制御システムの設計方法。

【請求項 2】 2つのコンポーネントウェア間を接続させる操作をする時に、データを出力する側のコンポーネントウェアの出力制約条件が、入力側の制約条件と同じか、または厳しい場合は接続を許すようにコンポーネントウェアを誘導し、そうでない場合はそれらのコンポーネントウェア間の接続を拒絶する、ことを特徴とする請求項 1 記載のネットワーク制御システムの設計方法。

【請求項 3】 設計時に 2つのコンポーネントウェア間を接続する際、コンポーネントウェア間でデータを試験的にやりとりし、データ出力側のコンポーネントウェアから出力されるデータに対して制約情報を取得し、この取得したデータの制約条件によって、データ出力側のコンポーネントウェア自身の出力制約情報をより厳しくし、対象のコンポーネントウェアと接続できるようにするツールを用意したことを特徴とする請求項 1 記載のネットワーク制御システムの設計方法。

【請求項 4】 ネットワーク管理システムにおける管理対象となる機器の http サーバが出力する HTML ファイルを、機器を管理するシステムがトラップする手段と、

そのトラップした HTML ファイルを、機器の現状における接続・設定の状態に照らして一部改変または最初から初期設定値を追加する等の処理を行う HTML 処理手段と、

処理後の HTML を、そのネットワークシステムの管理者用のブラウザに転送する転送手段とを有することを特徴とする請求項 1 記載のネットワーク制御システムの設計方法。

【請求項 5】 コンポーネントウェアまたは通信プロトコルの都合によって、設計時及び運用時に、本来 1 つのデータオブジェクトとして処理すべきデータを、適切に分割・走査して、個々のデータを適切なコンポーネントウェアに渡して処理をさせることを特徴とする請求項 1 乃至 4 いずれか記載のネットワーク制御システムの設計

方法。

【請求項 6】 設計時に接続するコンポーネントウェア群の中に、運用時にはインスタンスが 1 つしか存在しないコンポーネントウェア群と複数個存在するコンポーネントウェア群との分界点となる分界点コンポーネントウェアがあり、

設計時のシミュレートを行う際に、この分界点コンポーネントウェアを境に、分界点コンポーネントウェア自身が、分界点のコンポーネントブロックのインスタンスを多数作成・初期化し、

分界点コンポーネントウェアが複数個側へ適切にデータの流れを振り分け処理し、

ネットワークにおける 1 : n の関係をシミュレートするツールを用意することを特徴とする請求項 1 乃至 5 いずれか記載のネットワーク制御システムの設計方法。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、ネットワークを構成する各種の通信装置（ネットワークエレメント：NE）を保守・運用・管理するネットワーク制御システムの設計方法、特にコンポーネント指向を用いたネットワーク制御システムの設計方法に関するものである。

【0002】

【従来の技術】 ソフトウェアは一から作る時代から、組み合わせる時代になりつつある（青山幹雄：「ソフトウェアビッグバン」、情報処理、39 巻 4 号、pp. 338～341、1998）。通信網管理システムでも、コンパイルせずにそのまま利用できるプラグ&プレイ型ソフトウェア部品（コンポーネントウェア）を用意し、その組み合わせでシステムを構築することができれば、開発期間、コスト共に減少させることができる。Tel eManagement Forum における SMART TMN (TeleManagement Forum SMARTTMN Work Project, <http://www.tmforum.org/pages/smart.html>) 等でもその方向を目指しているが、コーディングレスに達するまでにはまだ技術的に解決しなければならない課題が多い。

【0003】 CORBA (OMG IDL Ver. 2. 2 Feb-1998)、DCOM (古山一夫「DCOM ガイドブック」、オーム社、1997) や、Java Beans (Java Beans™ API specification 1. 01 SUN Microsystems, 24-Jul-1997) 等の有力な枠組みが業界標準となり、インターフェースリポジトリやイントロスペクション等の仕様獲得機能が限定的ながら利用可能になったことにより、異なる組織で作られたコンポーネントの結合や、ツール製作者とコンポーネント製作者との分業が可能になってきている。

【0004】 ネットワーク制御システム (OpS) は通信ネットワークを構成する各種の通信装置を保守・運用

・管理する。このOpSの設計方法として、コンポーネントウェアを用いる方法が、依田育生・矢田浩二「コンポーネント指向通信網管理システム」(1998年電子情報通信学会ソサイエティ大会予稿集B-14-18、pp. 565、(社)電子情報通信学会、1998. 9. 7発行)、川幡太一・依田育生「通信網管理システムデータ変換部品の設計と実装」(1998年電子情報通信学会ソサイエティ大会予稿集B-14-19、pp. 566、(社)電子情報通信学会、1998. 9. 7発行)、前大道浩之・後藤哲明・多胡誠久「通信網管理システム用プロトコル部品の設計と実装」(1998年電子情報通信学会ソサイエティ大会予稿集B-14-20、pp. 567、(社)電子情報通信学会、1998. 9. 7発行)、川幡太一・矢田浩二「コンポーネント指向OpSのプロトタイプ」(1998年電子情報通信学会ソサイエティ大会予稿集B-14-21、pp. 568、(社)電子情報通信学会、1998. 9. 7発行)で提案されている。

【0005】図1はOpSの代表的な構成の概要を示すもので、同図(a)は階層型のシステム、同図(b)はコンポーネント型のシステムをそれぞれ示している。

【0006】コンポーネント型のシステムの設計には、コンポーネントウェア(CW)と呼ばれる部品が用いられる。

【0007】図2は従来のコンポーネント型のOpSの設計の流れを示すもので、パレット1上に事前に複数種類の汎用的なコンポーネントウェア(コンポーネント)2が用意されており、これらのうち、実際にOpSを作成するのに必要なコンポーネント2を選択し、該選択したコンポーネント2を適宜、カスタマイズし、それらを接続することによって作成していた。

【0008】この方法はまた、プロトコルの差異をコンポーネント化することにより、プロトコルの多様化の問題も解決した。

【0009】図3は前述したOpSの設計におけるデータの設定のようすを示すもので、コンポーネント2における各種のデータの設定はプロパティシート3と呼ばれる設定領域を開き、そこに各データ固有の表現方法で値を記述することにより行われる。また、各コンポーネント2間でやりとりされるデータを表示させ、モニタ(4)することもできるが、この表示も各データ固有の表現方法で行われる。

【0010】このようなOpSでは、図4に示すように、ユーザ5はアプリケーション6を介してネットワークエレメント(NE)7を操作する。ネットワークエレメント7を管理するアプリケーション6は、幾つものコンポーネント2の接続で構成され、ユーザはGUIコンポーネントであるボタン8及びグラフ表示9によって操作(GUI操作)する如くなっている。

【0011】ネットワークエレメント7からのイベント

は、アプリケーション6内のコンポーネント2間でやりとりして処理され、グラフ表示9を通してユーザへ伝えられ、また、ユーザの指示はボタン8を通してネットワークエレメント7に伝えられる。

【0012】

【発明が解決しようとする課題】(a)コンポーネント間でやりとりされる構造型データへの統一的なインターフェース

CMIP、SNMP等の伝統的な管理プロトコル、CORBA等の分散の枠組みに加え、HTML+Appletによるユーザインターフェース等、OpSが扱うプロトコルは多様化している。さらに、ルータ等のネットワーク機器では、詳細な設定はHTMLを用い、ブラウザ経由で行うものが増えており、これらは単体で管理する上では便利であるが、多数のネットワーク機器を集中的に管理したい場合、HTTPを新たな管理プロトコルとして捉え、機械処理する必要が生じている。

【0013】これらの多様な管理プロトコルを扱う場合、これまでは各管理プロトコルのプロトコルデータユニット(PDU)の構造を意識しながら、プログラムを製作する必要があった。例えば、CMIP、SNMPではASN.1を、CORBAではIDLを用いてデータが記述され、符号化方法や計算機言語へのマッピングもそれぞれ異なる。これらのプロトコル毎に専用のコンポーネント群を用意していくと、数多くのコンポーネントを設計しなければならないだけでなく、開発者にとっても考え方の異なる多くのコンポーネントを扱う必要が生じ、負担が増してしまう。

【0014】管理プロトコルを計算機言語で実装する際には、最も効率的に処理できるようにPDUの構造と計算機言語上のデータ構造のマッピングが図られているのが常識であり、効率の面からは既存のプロコルスタックで用いられているデータ構造を尊重すべきである。異なるプロトコルでコンポーネントを共有化するために、コンポーネント間で受け渡すデータを、安易に共通なデータ構造に変換すると、場合によってはプロトコルの処理に必須な情報が欠落して通信不能になる可能性がある。

【0015】コンポーネントウェアを指向したネットワーク制御システムの設計方法では、デバッグや値の入力、コンポーネントのプロパティシートにおける値の設定が非常に面倒であったため、これらを改良する必要が認められた。

【0016】(b)集中ネットワーク管理における管理対象機器の持つGUIインターフェースの利用

今後、ネットワーク管理で使用されるユーザーインターフェースは、開発コストがかさむアプリケーションではなく、安価なhttp+appletが主流になるとと思われる。また、管理対象となる機器も、httpによる管理手法を提供するようになっている。そのため、多数

のネットワーク機器の一括管理はこれまでの方法で行うとしても、個々の機器を管理する際には、それらの機器が提供する管理手法を最大限活用することが望ましい。

【0017】(c) データタイプと誘導

ソフトウェア部品の再利用を困難にしている原因の1つは、部品間で引き渡すデータタイプの不一致である (Gordon S. Novak Jr. "Software Reuse by Specialization of Generic Procedures through Views", IEEE Trans on Software Engineering, vol. 23, No. 7, pp. 401~417, July, 1997)。

【0018】一方で、コンパイラ等によって検査される強力な型付けは、開発者を適切に誘導し、プログラムの誤りを防ぐ有力な手段となっている (NMF040, 41, 42, 43 TMN/C++API, 1997)。通信網管理で扱うプロトコルのデータ構造は複雑になる場合が多く、このような誘導は必須である。

【0019】ところが、このデータタイプは情報定義毎に異なるのが一般的で、任意に結合可能な、汎用のコンポーネントを設計しようとすると、型付けを廃してデータタイプの抽象化を図る必要が生じる。これでは実行時にデータが渡せるかどうかを開発者が自分で調べることになり、ますます開発者の負担が増えるため、型付けに代わる誘導の仕組みが必須となる。

【0020】(d) コンポーネント間でやりとりされるデータの走査

従来技術では、コンポーネント間でやりとりされるデータは、一括してまとめて受け渡しを行っていた。そのため、HTMLデータや、CMIPのLinked Reply, ASN. 1のset of等のように、全体としては1つのデータだが、個別の要素に分解して個別に処理をしたい場合には、従来の手法では対応が困難だった。

【0021】(e) ネットワーク上における、1:nの関係性をコンポーネントで構築する際の問題

従来のコンポーネント技術の利点の1つとして、コンポーネントをツールで組立てながら動的に実行して動作を確認することが可能なことが挙げられる。しかし、この技術をOpS構築に適用しようとすると、OpSの中でしばしば現れる、マネージャ・エージェント、またはサーバ・クライアントのような1:nの関係では、nの方の状態が不特定多数個出現する。そのため、このような関係をツール上で動的にシミュレートすることが困難であった。

【0022】

【課題を解決するための手段】(a) コンポーネント間でやりとりされる構造型データへの統一的なインターフェース

実際にコンポーネント間でやりとりされる様々なデータ (ASN. 1やXML等の多様なデータ構造) について調べて、コンポーネント間でやりとりされる構造型データに対して統一的なツリー構造のインターフェースを定めた。これによって、データ自体は異なりつつも、統一されたツリー型データアクセスインターフェースでアクセスできるようにする。例えば、管理システムで使われるCWの各種設定をツリー構造メニューで行うことが可能になったり、コンポーネント間でやりとりされるデータのモニタをする際、そのデータを統一された形式で分かり易く見るできるようになった。

【0023】(b) 集中ネットワーク管理における管理対象機器の持つGUIインターフェースの利用

個々のネットワーク機器を接続した際に、機器のhttpサーバが出力するHTMLファイルを、機器を管理するシステムが管理データとして取り込む。このデータに対して、機器の現状における接続・設定の状態に照らし、一部分改変、または最初から初期設定値を追加する等の処理を行う。その処理後、データを再びHTMLに直して、そのネットワークシステムの管理者用のブラウザに転送する。

【0024】(c) データタイプと誘導

コンポーネント間での接続を適切に誘導するために、コンポーネントがやりとりできるデータに対して制約を導入し、この制約に基づいて、コンポーネント間の接続の可否 (誘導) を決めることにする。この接続の誘導には、以下の2つの方法がある。

【0025】(1) 各コンポーネントから、出力・入力データの型情報とは別に、型が取りうる値についての情報 (制約情報) を取得する。そして、データ出力側コンポーネントの出力制約条件が、データ入力側コンポーネントの入力制約条件に比べて同じか、または厳しい場合に限って、接続を許すようにコンポーネントを誘導する。

【0026】(2) コンポーネントだけではなく、コンポーネントから入出力されるデータに対しても、制約情報を取得できるようにする。この場合、出力側コンポーネントから出力されるデータの制約条件は、実際に、そのコンポーネントが動くまで分からない。このため、コンポーネントをツールで組立てている途中で、実際にデータを出力するコンポーネントまでとりあえずデータを流す。この流れてくるデータの制約条件によって、コンポーネント自身の出力制約情報をより厳しくし、入力側コンポーネントと接続できるようにする。即ち、この手法は「組立てながらシミュレートする」ことができないれば意味がない。

【0027】また、コンポーネント同士の接続判定を行う場合、データ出力コンポーネントの持つ出力制約条件が、データ入力コンポーネントの持つ入力制約条件に比べて厳しすぎるため、つながらないに関わらず、実際

に流れてくるデータは、出力コンポーネント自身の出力制約条件よりも厳しいというような場合がある。この時、見かけはコンポーネント同士の接続は無理でも、事実上はコンポーネント同士の接続が可能となる。ここで、実際にコンポーネント自身の接続を可能にするには、出力コンポーネントのみから出力制約情報を得るのではなく、出力コンポーネントから流れてくる出力データに対しても、出力制約情報を得る必要がある。

【0028】(d) コンポーネント間でやりとりされるデータの走査

巨大なデータを、複数のより細かいデータとして走査するようなコンポーネントを採用する。その際、走査して流す個々のデータに対して何らかのコンポーネントが処理を行った結果を、再び元の巨大データに反映させるコンポーネントや、そうしないコンポーネント等を用意する。

【0029】(e) ネットワーク上における、1:nの関係をコンポーネントで構築する際の問題

1:nの分界点におけるコンポーネントを境にして、分界点からn側のコンポーネントブロックのインスタンスを、分界点コンポーネントがシミュレート時に、多数作成・初期化する。また、サーバまたはマネージャ側のデータをクライアントまたはエージェント側に適切に振り分けたり、その逆を行う処理も分界点コンポーネントが行う。このようにして、ツール上でもネットワークの1:n関係を適切にシミュレートすることを可能にする。

【0030】

【発明の実施の形態】

【0031】

【実施の形態1】請求項1、2及び3に記載された発明が解決しようとする課題をまとめると、通信網管理のアプリケーションドメインにおけるソフトウェアのコンポーネント化を進めるに当たっては、コンポーネント間のデータ交換に対して以下が必要である。

【0032】1) 今後現れるより新しい管理プロトコルのPDU構造も包含できるような抽象的なデータ構造の設計

2) 抽象化によって失われる型付けに代わる新たな開発者誘導の仕組みの創出

3) プロトコル固有の実装の尊重

本実施の形態では、分散機能を持たない細粒度のコンポーネントを対象とすることを前提としている。即ち、各コンポーネントは同一プロセス内部で動作し、コンポーネント間のデータの交換はプロシージャ呼び出しの引数として渡すものとする。

【0033】同一プロセス内で結合するコンポーネント間で交換するデータは、一定の機能を持ったオブジェクト(イベントオブジェクト)とすることができる。上記の条件を満たすために、イベントオブジェクトには各管

理プロトコル固有のデータ構造をそのまま保持させると共に、プロトコルによらないアクセスインターフェースを実装させる。

【0034】図5は本発明のアクセスインターフェースの概要を示すもので、プロトコル固有の固有データ11にプロトコルによらないアクセスインターフェース12を実装させてイベントオブジェクト13を構成することによって、プロトコル固有のデータ構造を保持させたままコンポーネント14、15間で交換可能としている。

10 【0035】これ以降は、本発明のアクセスインターフェースの設計と実装の一例について述べる。

1. アクセスインターフェース

1. 1. PDUデータ構造の整理

通信網管理で使用する情報定義手法としては、GDMO/ASN. 1 (CMIP), MIB/ASN. 1 (SNMP), IDL (CORBA), DTD: Document type Declaration (HTML, SGML, XML《Extensible Markup Language (XML) 1. 0 W3C Rec. 10-Feb-1998. <http://www.w3.org/TR/RFC-xml>》、SQL (RDBMS) 等が考えられる。イベントオブジェクトの持つインタフェースは、これらの手法を用いて表現できるデータ構造の全てを操作できなければならない。また、可能な限り単純であることが望ましい。

【0036】まず、データ構造は基本型と、それを組み合わせた複合型との2つに大別できる。基本型は整数型や文字列型等、通常の計算機言語の原子型で表せるような型である。各プロトコルの定義手法において、基本型にはあまり差がない。基本型は計算機言語で直接的に表すことが可能であるため、アクセスインターフェースとしては、原子型毎に変更するメソッドと取得するメソッドを用意しておけば良い。

30 【0037】これに対して複合型は、各プロトコルで異なる表現方法が用いられている。本発明では、オブジェクト指向設計におけるオブジェクトの集合を表す、Collection (Dennis de Champeaux, Douglas Lea, and Penelope Faure, "Object-Oriented System Development", Addison Wesley, 1993) の考え方を採用し、分類を図る。

【0038】Collectionの種類として、図6に示すように、同一な要素を許す集合/許さない集合 (Bag/Set)、さらにBagの一種として順序付きの集合 (Sequence)、Setの一種としてkey及びvalueの組を要素として持つ集合 (Map) 等がある。

50 【0039】本発明では、異なる型を並べる複合型をMapに分類し、同じ型を並べる複合型をSequence

eに分類する方法を提案する。Mapに分類するものを構造型、Sequenceに分類するものを配列型と呼ぶことにする。

【0040】この外、Collectionの概念に当てはめることができない、プロトコルに特徴的なデータ型がある。これは、ASN.1におけるANY (Open Type)、IDLにおけるany等、実行時に決ま

型名	構造	説明
基本型	—	計算機言語の原始型であらわすことのできる型。
構造型	Map	複数の型を並べた型。
配列型	Sequence	同一の型を並べた型。可変長の配列。
ANY型	特殊	任意の型を内包できる型。

【0043】1. 2. 制約

データ構造をオブジェクトの集合として抽象化すると、従来のAPIにあった型による開発者の誘導が全くなくなる。それに代わる物として制約を導入する。制約は各データの型、取り得る値、要素数、要素の存在等の条件を表す情報でイベントオブジェクトから取得する。

【0044】さらに制約の応用範囲を広げ、データ構造の一部を制約によって表す。例えば、ASN.1のCHOICEや、IDLのunionは、複数の型の候補のうち、1つを選択して使用するという複合型であるが、これらの型を、構造型に対して、要素のうち1つだけが存在するという制約を設けたものと解釈する。即ち、SEQUENCEとCHOICE、structとunionに対しては、構造型上は同じインターフェースとし、制約によってその意味の違いを表す。このように、従来、構造型として分類していた差異を制約の違いと解釈することによって、インターフェースの種類を縮小することができる。

【0045】1. 3. 誘導情報・メタデータ

情報定義には、型名、構造型の要素の名前、数値につけられた名前等、実際に転送されるデータには含まれないが、開発者の理解を助ける情報が多く含まれている。ツールを用いて、的確に開発者を誘導するためには、これらの情報を制約情報等と共にツールがイベントオブジェクト自身から取り出せるようにしておく必要がある。

【0046】一方、イベントオブジェクトは通常、実行時に生成される一時的なオブジェクトであり、コンポーネントの組み合わせ時にはインスタンスが存在しないことが考えられる。そのため、インターフェースに関する

る型である。この型については、実行時の型情報の取得設定インターフェースが必要であり、この型をANY型と呼ぶことにする。

【0041】以上を整理すると、表1の4つの型になる。

【0042】

【表1】

情報はメタデータとして1つのオブジェクトにまとめ、データ構造の名前等からメタデータを検索できるようにしておく必要がある。

2. インタフェース定義

以上の考え方に基づいたアクセスインターフェースの骨格をJava (JDK1.2)で定義した場合の例を以下に示す。JDK1.2のCore APIにはCollectionが導入され、アプリケーションに因らない共通なインターフェースとして推奨されている。JDK1.2では、SequenceをListと呼んでいる。JDKのCollectionの重要な点は、インターフェースと実装を明確に分離しており、本発明のインターフェースをCore APIに準備された汎用のライブラリと区別なく使えることである。

【0047】また、同一Java Virtual Machine内でデータを交換するインターフェースとして、Info Bus ()があり、原子型の取得設定にはこれを用いる。Info Busでは交換データのAPIの1つとしてCollection APIを定めており、本発明のインターフェースはInfo Busの仕様を満たしたものとなっている。

【0048】2. 1. データインターフェース

表2にデータアクセスのために使用するインターフェースを示す。新規に定義しているのは、Pdu Data、Pdu Any Dataだけであり、それぞれ、表3、表4に示す。

【0049】

【表2】

型	インターフェース	説明
基本型	PduData, ImmediateAccess	ImmediateAccessを介して基本型データを取得設定する
構造型	PduData, Map	Mapのkeyには要素名を, valueには要素値を指定する
配列型	PduData, List	Listの要素に配列の要素を対応させる
ANY型	PduData, PduAnyData	PduAnyDataで内包するデータ, メタデータを取得設定する

【0050】

【表3】

戻り値	関数	説明
String	getTypeName()	タイプ名取得
PduMetaData	getMetaName()	メタデータ取得
Map	getRestriction()	制約情報取得

【0051】

【表4】

戻り値	関数	説明
void	setAnyMetaData(PduMetaData)	メタデータ設定
PduMetaData	getAnyMetaData()	メタデータ取得
Void	getAnyData(PduData)	データ設定
PduData	getAnyData()	データ取得

【0052】 2. 2. メタデータインターフェース
型情報は制約の一種であるが、型はオブジェクトの生成
削除に関わる基本構造であるため、制約情報から切り離
してメタデータに組み込む。表5、表6にメタデータに

アクセスするためのインターフェースを示す。

【0053】

【表5】

型	インターフェース	説明
基本型	PduMetaData, ImmediateAccess	ImmediateAccessを介して基本型データのクラスを取得
構造型	PduMetaData, Map	Mapのkeyに要素名を指定し、valueとして要素のメタデータまたはクラスを取得
配列型	PduMetaData, ImmediateAccess	ImmediateAccessを介して要素のメタデータ・クラスを取得
ANY型	PduMetaData	

【0054】

【表6】

戻り値	関数	説明
int	getKind()	基本型・構造型・配列型・ANY型の種別の取得
String	getProtocol()	プロトコル名の取得
String	getTypeName()	タイプ名の取得
PduMetaData	getParent()	親のメタデータの取得
Map	getRestriction()	制約情報の取得
PduData	newInstance()	データの生成

【0055】 2. 3. 制約・誘導情報
データに与えられる制約の種類は無限にあり、一般的な
記述方法はない。ASN. 1のSub type及びXMLのDTDで表現できる制約を完全に表現できること
を目標に、制約を整理すると表7になる。この表では、

制約情報をMapで表し、keyが意味を、value
が値を表す。また、誘導のための情報を表8に表す。

【0056】

【表7】

key	value	型	説明
value	値、範囲の集合	全	取りうる値
size	整数値、整数範囲の集合	基本 構造 配列	文字列の長さ 存在する要素数 要素数
regexp	正規表現	基本	文字列の正規表現
present	要素名の集合	構造	存在すべき要素名
absent	要素名の集合	構造	存在しない要素名
type	型名の集合	ANY	設定できるタイプ名
element	制約	基本 構造	文字列の文字の制約 要素に共通の制約
named Element	要素名と制約	構造	特定の要素の制約
element Sequence	制約の並び	配列	要素の並びの制約
or	制約の集合	全	各制約の論理和
and	制約の集合	全	各制約の論理積
not	制約	全	制約の反転

【0057】

【表8】

key	valueの型	型	説明
valueNames	名前と値のMap	全	各値に名前がある場合
indexNames	名前と整数値のMap	配列	配列型のインデックス値に名前がある場合
typeNames	名前とメタデータのMap	ANY	各タイプに名前がある場合
defaultValue	値	全	デフォルト値
defaultValues	名前と値のMap	構造 配列	要素のデフォルト値。配列型の場合には、名前にindex値を用いる

【0058】3. 実装と評価

本発明のアクセスインターフェースについて、ASN. 1及びXMLに対してJavaにおける実装を行った一例を示す。ASN. 1とXMLの表現力は似ており、1つの規格をASN. 1とXMLの前身であるSGMLの双方で記述している場合がある。ところが、通常の実装では全く異なったAPIが用いられるため、アクセスインターフェースの効果を試すのに最適な一例である。

【0059】ASN. 1に対しては、Java上に構築した独自の実装の上にアクセスインターフェースをかぶせる。XMLに対しては、XML及びHTMLのアクセスインターフェースとしてW3Cにより進められている

DOM (Document Object Model) (Document Object Model (DOM) Level 1 Specification Ver. 1.0 W3C Rec, 1-Oct-1998. <http://www.w3.org/TR/REC-DOM-Level-1>) による実装の上にアクセスインターフェースをかぶせる。

【0060】表9にASN. 1のタイプと本発明のインターフェースの対応を、表10にXMLの要素との対応を示す。

【0061】

【表9】

タイプ	型	制約・誘導情報
ENUMERATED	基本	value : namedNumber
INTEGER	基本	valueNames : namedNumber
SEQUENCE	構造	present : Optional でない要素名 defaultValues : デフォルト値
CHOICE	構造	size : 1
SET OF	配列	なし
BIT STRING	配列	element : Boolean indexNames : namedBit
OCTET STRING	配列	element : Byte
ANY	ANY	なし
Subtype	親の型	size : SizeConstraint element : Permitted Alphabet namedElement : Inner subtyping等

【 0 0 6 2 】

【表 1 0】

タイプ	型	制約・誘導情報
タグ名	基本	values : 上挿入可能なエレメント
CDATA	基本	regexp :]] > を含まない文字列
Document固有情報	構造	present : 省略できない部分
Document配下のノード群	配列	なし
Document配下のノード	構造	size : 1 absent : 許されないノード種類
Element固有情報	構造	present : 必須部 (タグ名等)
Element配下のノード群	配列	elementSequence : エレメント定義の構造
Element配下のノード	構造	size : 1 absent : 許されないノード種類

【 0 0 6 3 】 ASN. 1 では、データ構造が予め厳格に定められているのに対し、XML では文章構造を表す DTD の知識がなくとも Well-Formed な文書としてこれを取り扱う必要がある。DOM では文書の要素 (エレメント、属性、テキスト等) をノードとしたツリーで構造を表しているが、1 つのノードは、固有の情報 (エレメントのタグ名等) と配下のノードを持っており、かつ、DTD に依存しないよう、配下のノードには任意の種類のノードが設定可能であるため、図 7 に示すように、1 つのノードを 3 つのアクセスインターフェース (ノード固有情報を表す構造型、配下のノード群を表す配列型、配下のノードの種類を表す構造型) に分けて表すことにした。

【 0 0 6 4 】 図 8 は本発明によるデータの設定のようすを示すもので、各コンポーネント 2 に対する各種の設定は前述したアクセスインターフェースに基づく共通インターフェース 2 1 を介してツリー構造のメニュー 2 2 で行われる。

【 0 0 6 5 】 また、データ編集コンポーネントを製作し、ASN. 1、XML と同じコンポーネントで編集できることを確認した。この際、制約情報を適宜表示することによってある程度ユーザを誘導できることが分かった。ASN. 1、XML の各編集画面の一例を、図 9、図 1 0 にそれぞれ示す。

4. 制約のツール上における誘導への活用

請求項 2、3 に記載した本発明の O p S の設計方法を、図 1 1、図 1 2 にそれぞれ示す。

【 0 0 6 6 】 従来は、図 1 3 に示すようにエディタ上に置かれた 2 つのコンポーネント間の接続チェックに型のみを使用していた。しかし、コンポーネント間の接続は syntax な妥当性の上に、semantic な妥当性が問われるはずであり、コンポーネント自身が持つ制約情報は、semantic を完全とは言わないまでもかなりの部分を包含している。

【 0 0 6 7 】 そこで、図 1 1 では、2 つのコンポーネント間を接続する際に、互いの制約条件を確認するようにする。そして、出力側の制約条件が、入力側の制約条件に比べて厳しい場合は、接続処理を行う。一方、そうでない場合は、それらのコンポーネント間の接続を拒絶する。

【 0 0 6 8 】 さらに、図 1 2 では、例え出力コンポーネント自身の制約条件が緩くても、そこを流れるデータの制約条件が厳しければ、接続を許すようにする。

【 0 0 6 9 】 このように、図 1 1、図 1 2 で説明した手順を従来のコンポーネント間接続のツールに挿入することにより、コンポーネント間でやり取りするデータの矛盾から生じるバグを O p S の設計の段階で除去できる。

【 0 0 7 0 】

【実施の形態 2】 請求項 4 及び 5 に記載した本発明は、50 個々のネットワーク機器を接続した際における、機器

(ネットワークエレメント／NE) の設定に関係する。ここでは、ネットワークの管理対象となる機器 (NE) に対し、h t t p で制御し、その h t m l (またはXML等) 出力を管理システムが管理プロトコルデータとして扱い、それを走査して必要な部分の修正を請求項5に記載した発明で行い、その結果を管理システムのユーザーに h t m l として送出することによって、個々の管理機器が提供する操作方法を最大限に活用するという請求項4の発明を説明する。

【0071】図14はこれらの発明の手法を用いている実施の形態の一例を示すもので、31はネットワーク、32はネットワークエレメント (NE)、33は請求項4に記載した各手段、即ちネットワーク管理システムにおける管理対象となる機器の h t t p サーバが出力するHTMLファイルを、機器を管理するシステムがトラップする手段と、そのトラップしたHTMLファイルを、機器の現状における接続・設定の状態に照らして一部改変または最初から初期設定値を追加する等の処理を行うHTML処理手段と、処理後のHTMLを、そのネットワークシステムの管理者用のブラウザに転送する転送手段とを構成するマネージャ、34はユーザが操作するGUI、例えばWWWブラウザを備えたPCである。

【0072】ネットワーク31はネットワークエレメント32が複数接続されて構成されており、ネットワーク制御システム (OpS) はこれらの機器を管理する。ここでは、管理対象となる各ネットワークエレメント32は h t t p サーバを備えているとする。これらのネットワークエレメント32は、SNMP等の一般的なネットワーク管理プロトコルによって一括的に管理することも可能であり、また、個々の機器に関しては、HTTPプロトコルによって、GUI34を活用して、それらの機器にとって最善の形、即ちこれらの機器が提供するGUIインターフェースを尊重する形で機器を管理することも可能である。

【0073】ここで、個々のネットワーク機器を、ネットワーク管理システムのユーザが、h t t p プロトコルを使って設定したいとする。すると、ユーザは、h t t p リクエストを、マネージャ33を経由して各機器に対して送出する。すると、この場合、マネージャ33は、各管理対象機器が送出する h t t p の内容を最大限に尊重しつつ、それらの中でネットワークシステム全体に関する設定項目に対しては、マネージャ33が最初から設定しておく形で、ユーザに h t m l またはそれに類するデータを送るのが望ましい。そのため、一度送られてきた h t m l データを、管理データとしてシステムの中で処理し、そのデータの中の必要な部分に対しては適宜予め定められた方法によって、データの設定等の処理を行い、その結果となるデータを、再び h t m l データに直して、ユーザに送出する。

【0074】このようにすると、ユーザは、各機器から

送出されるであろう多数の設定項目のうち、予めネットワークシステム全体に関わる部分は最初から設定された形で設定項目を提示されるため、ネットワーク機器の設定が簡易化し、設定の負担が軽くなり、また、設定の誤りによってネットワーク全体に障害が及ぶのを防ぐことが可能になる。これが、請求項4で述べた発明である。

【0075】次に、これをコンポーネント指向の枠組みで実現する手段として、請求項5で述べられた方法をどのように適用するかを説明する。

【0076】請求項5の発明は、図15に示すように、イベントの通知が巨大なデータオブジェクト41で伝わってきた際に、それをより細かい複数のデータオブジェクト42単位に走査する処理を行うコンポーネント43である。

【0077】図15の左側からオブジェクト41がイベントとして次段のコンポーネントへ伝わっている。このイベント通知で、データを複数のより細かい部分に走査するコンポーネント43が、入力された巨大なデータオブジェクト41をより細かいデータオブジェクト42として出力する。細分化することで、次段のコンポーネントは小さなデータオブジェクト単位に処理することが可能になる。その処理結果は、必要ならば、もとに反映させることが可能となる。

【0078】ここで断っておくが、従来の通信プロトコルデータは、全て1つのPDUに収まることを仮定していた。しかしながら、HTMLの構造型データやCMIPのLinked Reply、ASN.1のset of等、個々の要素の操作をコンポーネントで行いたい場合は、このような従来の手法をそのまま適用することは困難である。そのため、図15で説明した複数データオブジェクトに分割するコンポーネントの採用等、何らかの形でデータオブジェクトデータを細切れにした形でも処理ができるようなコンポーネントを考えて準備しておくことが重要である。

【0079】ここで、図16を用いて、請求項4の発明を、請求項5の発明でどのように実現するかを具体的に説明する。まず、NE51から、HTTPプロトコルリクエストの返答として、HTTPプロトコルレスポンスによって、HTMLデータがマネージャ52に返ってくるから考える。この時、マネージャ52内のHTTP受信コンポーネント53は、HTMLデータを他の通信プロトコルのデータと同様に、コンポーネント間でやりとりされる形のツリー上の構造データに変換して、それを出力先のコンポーネントに送り出す。ここで、このHTTP受信コンポーネント53は出力データを、請求項5で述べた、データ走査コンポーネント54に入力として渡すとする。

【0080】データ走査コンポーネント54は、渡されたツリー構造型のデータを、個々に走査 (分解) して、個々のデータとして、例えばHTMLを調べて適宜必要

な設定項目について設定を行うコンポーネントに渡す。FORM設定処理コンポーネント55は、次々と細切れになって送られてくる各入力データについて逐次調べ、それがカスタマイザで指定されたタイプのFORMデータであれば、それを必要に応じて変更する。このようにして、大きい構造型データの各部分について、処理を行うことが可能になる。

【0081】このようにして処理されたデータを、データ走査コンポーネント54は、マネージャ52を操作するユーザへのHTTP（送信）コンポーネント56に送出する。HTTPコンポーネント56は、受け取った構造型データを再びHTMLに直して、ユーザのGUI57に送る。

【0082】こうすることによって、ユーザはNE51から送られてくるデータのうち、マネージャ52が必要に応じて修正した部分を除いて、ほぼNE51が提供するGUIをそのまま活用する形で利用することが可能になるのである。

【0083】

【実施の形態3】従来のコンポーネント技術によるOpS構築手法では、設計時には、サーバ・クライアントや、マネージャ・エージェントのような、運用時に1:nの動的な関係になるようなコンポーネントの接続関係でも、1:1の静的に接続されていた。しかしながら、実際には、これらマネージャ・エージェントの接続は常に設計当初のままの1:1の静的な関係ということではなく、場合によっては、1:nの関係となる。これには、エージェントの増設（新設したエージェントを同じマネージャに接続し追加する）や、逆に一部を撤去（稼働していたエージェントをマネージャから切り離す）するような場合も含まれる。同様の動的な接続・切り放しの関係は、サーバ・クライアント間でも起りうる。

【0084】ツールで接続したら、その場ですぐにその接続を実行したり、シミュレートしたりできるのが、コンポーネント指向の特徴であった。しかしながら、このような1:nの関係は、ツール上でシミュレートするのは困難であった。その解決方法として、1:nの分界点となるコンポーネントが、自身でn個側のインスタンスをカスタマイザの設定に応じて生成したり、または生成先に対して適切に送出したりすることによって、この関係をシミュレートする。

【0085】この仕組みは、請求項6記載の発明のOpS設計方法では、図17に示すように、コンポーネントをマネージャとエージェントとの間にまたがる2つのコンポーネントに分離するようにする。OpS設計ツール60上ではコンポーネント61から65までが接続されている。このうち、OpS設計の当初からコンポーネント61はクライアントで、65はNEである。また、コンポーネント62、64はネットワーク上での通信を示すコンポーネントである。

【0086】実際のOpSが運用される時には、マネージャ（またはサーバ）66にコンポーネント62、63、64が移され、さらに、そのコンポーネント62、64は分離（62'、64'）してクライアントとエージェントにも移される。

【0087】こうすることで、クライアント側やエージェント側が変わったり、複数のクライアントやエージェントがサーバまたはマネージャと接続し合っても、対応可能となる。図18はNEを動的に登録した後の結合関係を示すものである。

【0088】本発明では、運用時における、このようなネットワーク上での1:nの関係を、設計時に仮想的にコンポーネント61、65のインスタンスをコンポーネント62や64が増やし、また、コンポーネント62や64がコンポーネント63からのデータを適切に振り分けたり、その逆を行うことによって、この状況を適切にシミュレーションできることを示す。

【0089】多様な主信号を扱うマルチメディアネットワークでは、様々な管理プロトコルが使用される。本発明では通信網管理システムをコンポーネント指向で製作することを提案し、異なる種類のプロトコルデータを同一のコンポーネントで取り扱うためのアクセスインターフェースを示した。また、Collectionの概念と制約情報を用いて、プロトコル種別に依存しないインターフェースを定義し、ASN.1やXML等の異なるプロトコルのデータに対し、それらの構造を維持したまま、汎用的なコンポーネントがデータの処理を維持できることを示した。本発明のインターフェースは、プロトコル固有のAPIが全く異なる両者に適用可能で、同一のデータ編集コンポーネントで双方の操作が可能である。

【0090】実施の形態では、本発明による通信網管理システムをコンポーネント指向で製作する場合に必要な、プロトコルデータをコンポーネント間で交換するためのインターフェースについて述べた。Collectionの概念と制約情報を用いて、プロトコル種別に依存しないインターフェースを定義し、ASN.1とXMLに対して実際にJavaを用いてインプリメントを行う。両者のプロトコル固有のAPIは全く異なっているが、本発明のインターフェースは双方に適用可能である。本発明のインターフェースを介してデータの編集を行うGUIコンポーネントは双方に適用でき、プロトコルに依存することなくデータを操作できる。

【0091】

【発明の効果】以上説明したように、請求項1の発明によれば、多様なデータ構造を同時に統一されたデータアクセスインターフェースでアクセスすることを、ツリー型エディタで実現できる。この結果、ユーザがOpSを設計するのに必要となるコンポーネントウェア（CW）を設定することが容易となり、CW間でやり取りされる

データも視覚的な理解が可能になる。

【0092】また、請求項2, 3の発明によれば、コンポーネント間での接続を適切に誘導するために、コンポーネントがやりとりできるデータに対して制約を導入し、この制約に基づいて、コンポーネント間の接続の可否(誘導)を決めた。このようにすることで、OpS設計で誤った接続を解消でき、設計ミスを削減できる。

【0093】また、請求項4記載の発明によれば、多数のネットワーク機器が接続されている場合、その中の1つの機器の設定変更を行うGUIについては、個々の機器が搭載している設定用GUI／httpサーバを活用し、変更に伴うネットワーク全体の調整を図るためにマネージャを活用することが可能になる。

【0094】また、請求項5記載の発明によれば、1つのデータオブジェクトに対して、その構造の中身について個々の処理をコンポーネントが行いたい場合、そのデータオブジェクトに対して走査・分解を行うコンポーネントを導入し、その使用によって、例えば請求項2, 3の発明をコンポーネント指向の枠組みのなかで行うことを示した。

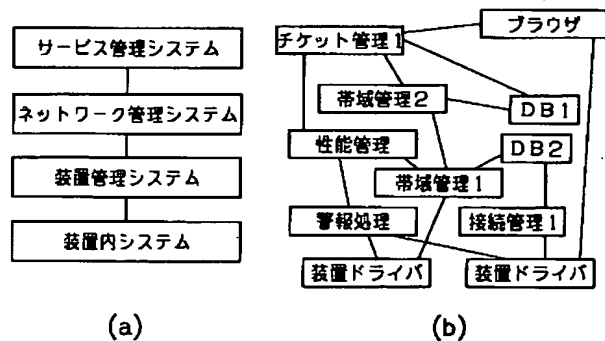
【0095】さらにまた、請求項6記載の発明によれば、1:nの分界点におけるコンポーネントを境にして、分界点からn側のコンポーネントブロックのインスタンスを、シミュレート時に、分界点のプロトコルを表すコンポーネントが多数作成・初期化する。また、それらを分界点コンポーネントが適切にデータの流れの振り分け処理をする。このようにすることによって、ツール上でも、ネットワークの1:n関係を適切にシミュレートすることを可能にした。

【図面の簡単な説明】

【図1】ネットワーク制御システムの代表的な構成図

【図2】従来のコンポーネント型のネットワーク制御シ

【図1】



ステムの設計の流れを示す図

【図3】OpSの設計におけるデータの設定のようすを示す図

【図4】ネットワークエレメントに対する操作のようすを示す図

【図5】本発明のアクセスインターフェースの概要を示す図

【図6】複合型のデータ構造として用いるCollectionの種類を示す図

10 【図7】DOMとアクセスインターフェースとの対応を示す図

【図8】本発明によるデータの設定のようすを示す図

【図9】ASN.1データの編集画面の一例を示す図

【図10】XMLデータの編集画面の一例を示す図

【図11】請求項2対応のコンポーネント間の接続の処理フロー

【図12】請求項3対応のコンポーネント間の接続の処理フロー

【図13】従来のコンポーネント間の接続の処理フロー

20 【図14】請求項4, 5にかかわる実施の形態の一例を示す図

【図15】巨大なデータオブジェクトの処理のようすを示す図

【図16】請求項5にかかわる処理の一例を示す図

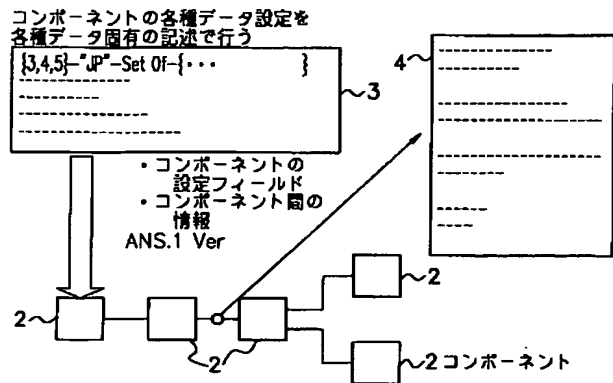
【図17】請求項6対応の実施の形態の一例にかかわるNEの動的な登録のようすを示す図

【図18】請求項6対応の実施の形態の一例にかかわる登録した後の結合関係を示す図

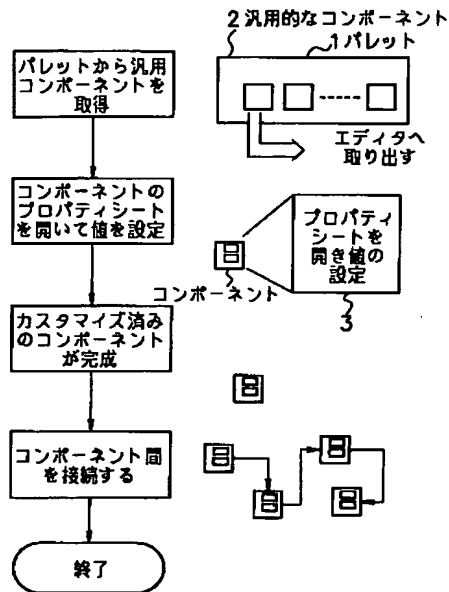
【符号の説明】

30 2, 14, 15: コンポーネント、11: 固有データ、12: アクセスインターフェース、13: イベントオブジェクト、21: 共通インターフェース。

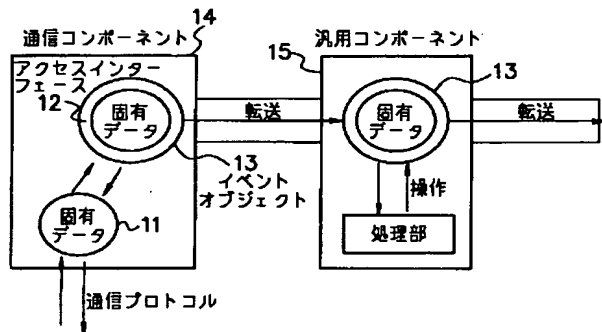
【図3】



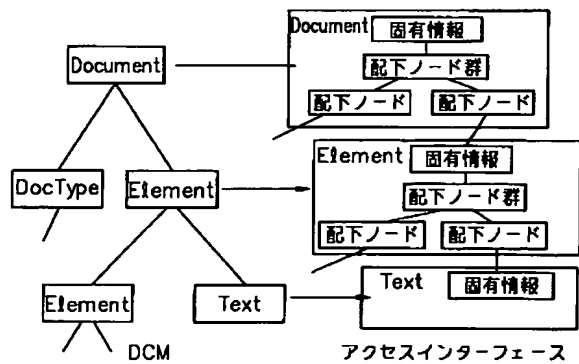
【図 2】



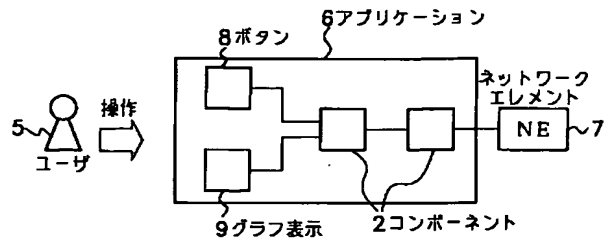
【図 5】



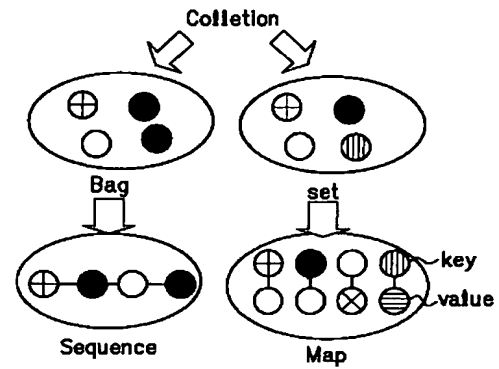
【図 7】



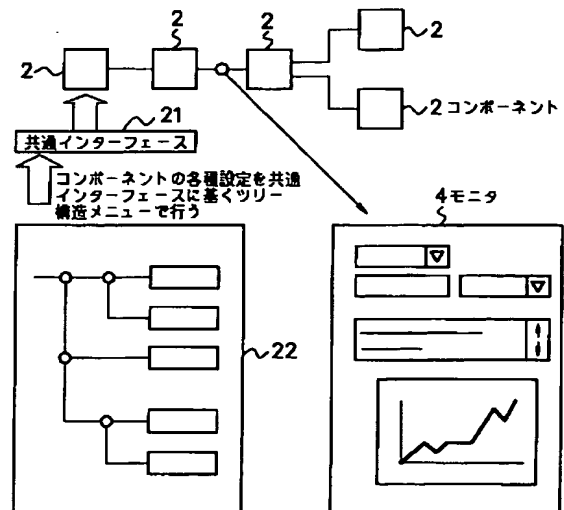
【図 4】



【図 6】



【図 8】



【図 9】

land: {item: equality: {attributeId gobalform: {2 9 3 2 7 60}, attributeValue gobalform: {1 9 3 2 3 13}}, item: eq

♀ □ {item: equality: {attributeId gobalform: {2 9 3 2 7 60}, attributeValue gobalform: {1 9 3 2 3 13}}, item: eq

♀ □ {item: equality: {attributeId gobalform: {2 9 3 2 7 60}, attributeValue gobalform: {1 9 3 2 3 13}}

♀ □ equality: {attributeId gobalform: {2 9 3 2 7 60}, attributeValue gobalform: {1 9 3 2 3 13}}

♀ □ {attributeId gobalform: {2 9 3 2 7 60}, attributeValue gobalform: {1 9 3 2 3 13}}

♀ □ gobalform: {2 9 3 2 7 60}

♀ □ {2 9 3 2 7 60}

♀ □ 2

♀ □ 9

♀ □ 3

♀ □ Item:

♀ □ eq

♀ □

♀ □

Information

--- Current type ---

CMIP-1.ObjectInstance

--- Restrictions ---

SIZE(1)

--- Current value ---

distinguishedName: {{{attribute Type{2 5 4 6}, attributeValue "JP"}}, {{attribute Ty

Elements

distinguishedName [2] IMPLICIT CMIP-1, DistinguishedName

nonSpecificForm [3] IMPLICIT OCTET STRING

localDistinguishedName [4] IMPLICIT CMIP-1.RDNSequence

Operation

edit

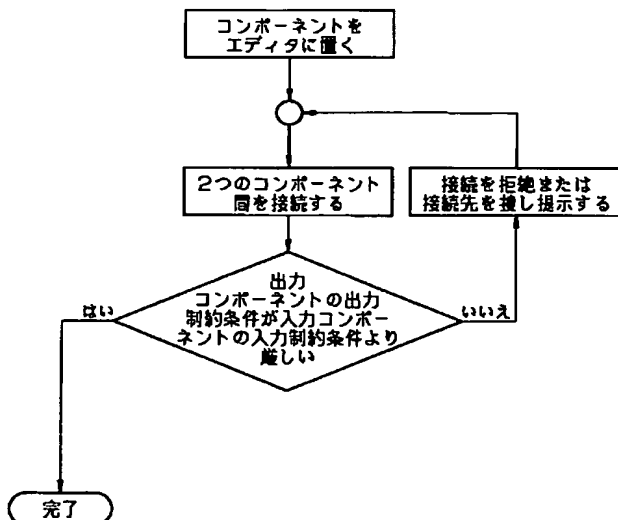
remove

変更 戻る

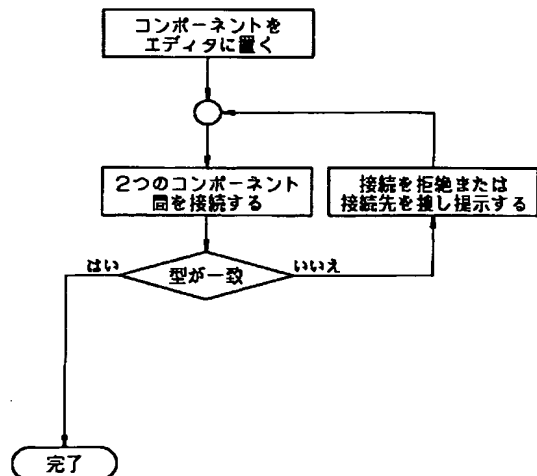
♀ □ name: "SampleManager"

♀ □ "SampleManager"

【図 11】



【図 13】

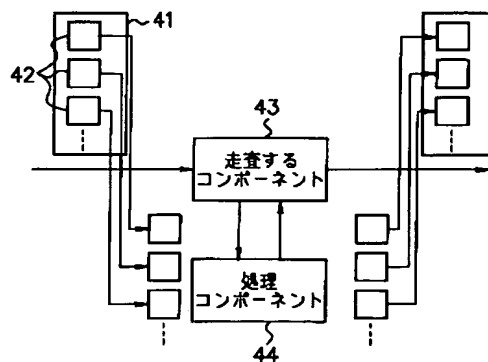


The screenshot shows the XML Editor window with the following content:

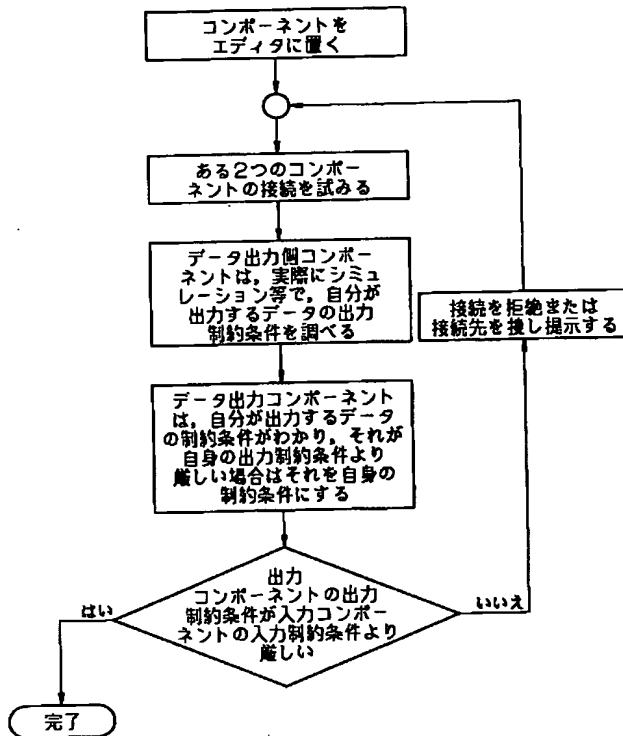
- Information Tab:**
 - Restrictions---
 - SIZE(1)
 - Current value ---
 - <computer id="nobunaga">
 - <spec><os>Solaris 2.6</os><cpu>Super Sparc</cpu></spec>
 - <address>nobunaga.onlab.ntt.co.jp</address>
- Elements Tab:**
 - *element #element
 - text class java.lang.String
 - ignorable class java.lang.String
 - entryRef class java.lang.String
 - cdata class java.lang.String
 - p1 #processing-instruction
- Operation:**
 - edit
 - remove

Buttons at the bottom: 変更 (Change), 戻る (Back).

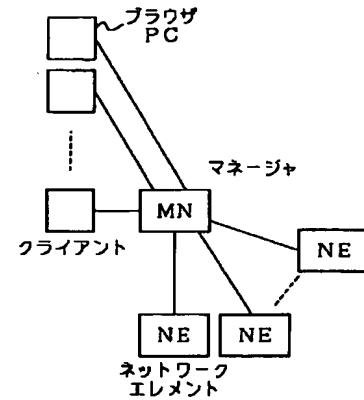
【図 15】



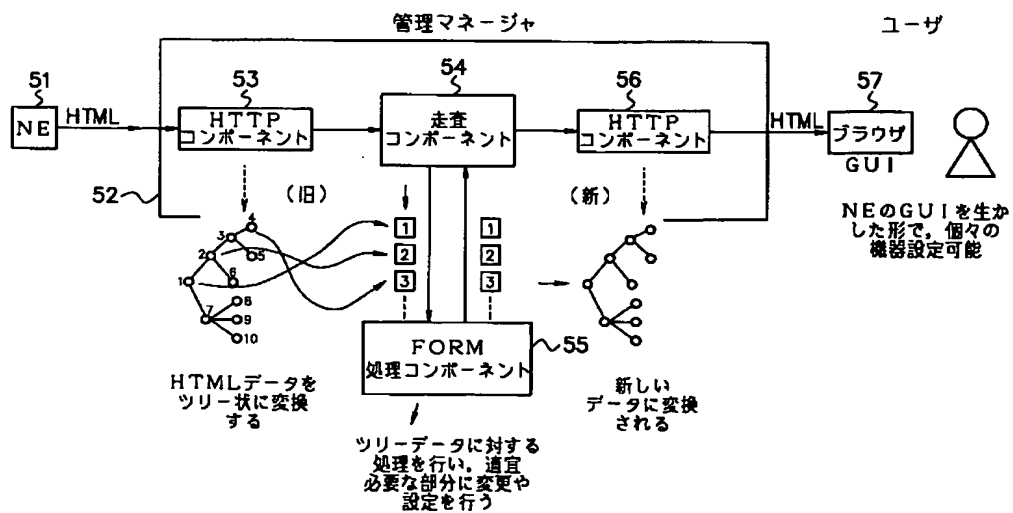
【図 12】



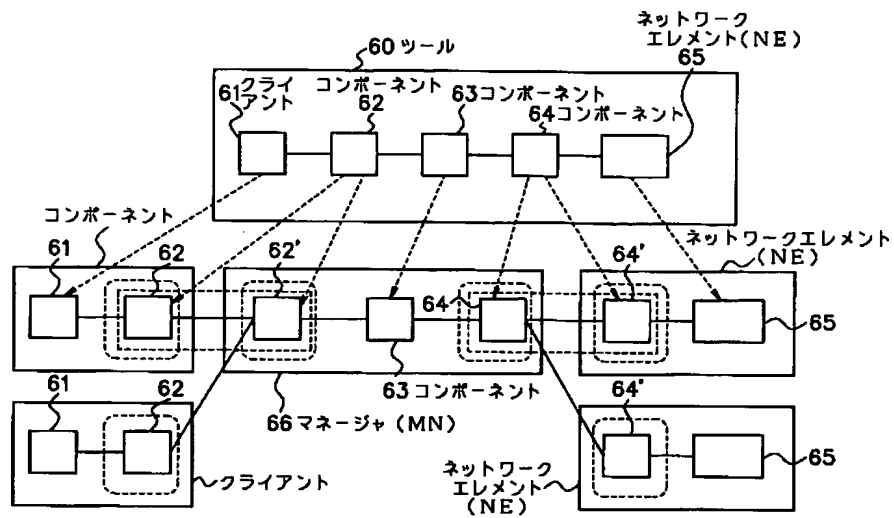
【図 18】



【図 16】



【図17】



フロントページの続き

Fターム(参考) 5B076 DD05

5K034 AA06 AA18 FF01 FF11 FF15

FF18 HH14 HH17 HH18 HH26

JJ23 MM39